UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/593,262	09/18/2006	Yves Gattegno	200800984-6	6443

22879 7590 06/23/2011

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
FORT COLLINS, CO 80528

EXAMINER	
PATEL, SHAMBHAVI K	
ART UNIT	PAPER NUMBER
2128	

NOTIFICATION DATE	DELIVERY MODE
06/23/2011	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

JERRY.SHORMA@HP.COM
ipa.mail@hp.com
laura.m.clark@hp.com

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/593,262
Filing Date: September 18, 2006
Appellant(s): GATTEGNO, YVES

Christopher R. Rogers
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 03/22/11 appealing from the Office action mailed 11/26/10.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The examiner has no comment on the appellant's statement of the status of amendments after final rejection contained in the brief.

(4) Status of Amendments After Final

The examiner has no comment on the appellant's statement of the status of amendments after final rejection contained in the brief.

(5) Summary of Claimed Subject Matter

The examiner has no comment on the summary of claimed subject matter contained in the brief.

(6) Grounds of Rejection to be Reviewed on Appeal

The examiner has no comment on the appellant's statement of the grounds of rejection to be reviewed on appeal. Every ground of rejection set forth in the Office action from which the appeal is taken (as modified by any advisory actions) is being maintained by the examiner except for the grounds of rejection (if any) listed under the subheading "WITHDRAWN

REJECTIONS." New grounds of rejection (if any) are provided under the subheading "NEW GROUNDS OF REJECTION."

(7) Claims Appendix

The examiner has no comment on the copy of the appealed claims contained in the Appendix to the appellant's brief.

(8) Evidence Relied Upon

20030200290	Zimmerman	10-2003
6857069	Rissmeyer	2-2005

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 1-46 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Zimmerman (US Pub. No. 2003/0200290)** in view of **Rissmeyer (US Patent No. 6,857,069)**.

Regarding claim 1:

Zimmerman discloses a method, performed by a suitably programmed computer, for software emulation of hard disks of a data processing platform at the level of the operating system with parameterizable management of requests for writing and reading data, the method comprising:

a. creating a representation of a real hard disk wherein the location for loading and execution of components of the operating system of the data processing platform may be modified **([0019]: drives hold O/S; [0058]: drives may comprise any nonvolatile storage devices)**

b. loading on said data processing platform one or more peripheral drivers **([0044]: storage driver and network filter driver)**, wherein at least one of the peripheral drivers

communicates with a data storage support containing the data of the representation of the real hard disk **([0044]: storage driver and network filter driver used to download files from the virtual drive).**

c. simulating the behavior of the real hard disk for the operating system wherein the method transforms programming contained on the real hard disk into an emulated hard disk **([0026]: "virtual" storage device)** capable of controlling read and write operations on a client station and among two or more client stations **([0026]: requests for data result in transparent emulation of operations of local disk at each of the clients; [0022]: one or more client devices)**

Zimmerman does not explicitly disclose modifying the sequence for loading and executing of components of the operating system of the data processing platform may be modified. **Rissmeyer teaches** modifying the sequence for loading and executing of components of the operating system of the data processing platform may be modified **(column 1 lines 46-59: loading a network driver before loading a disk driver in an OS booting over a network).** At the time of the invention, it would have been obvious to one of ordinary skill in the art to combine the teachings of Zimmerman and Rissmeyer because this allows the traditional controlling of devices prior to booting to occur without customized parts **(Rissmeyer: column 1 lines 11-44).**

Regarding claim 2:

Zimmerman discloses the method of claim 1 wherein the management of said data write requests that the operating system sends to the emulated hard disk is accomplished at the peripheral driver level **([0044]: drivers),** the written data being stored according to the parameterization of the peripheral drivers, and wherein the parameterization accommodates the written data being stored in the memory accessible to the OS using the emulated hard disk **([0069]: writes locally cached).**

Regarding claim 3:

Zimmerman discloses the method as claimed in claim 1 wherein the management of said data read requests that the operating system sends to the emulated hard disk is accomplished at the peripheral driver level (**[0044]: storage driver and network filter driver used to download files from the virtual drive),** the readings of previously written data being performed in the nonvolatile storage space accessible to the operating system using the emulated hard disk **([0058]: nonvolatile)**

Regarding claim 4:

Zimmerman discloses the method as claimed in claim 1 wherein the emulation of the hard disk is accomplished by the agency of a single monolithic peripheral driver which communicates with the OS in the manner of a hard disk and which communicates with the support containing the data of said emulated hard disk in a manner specific to this support **([0055]: single driver for some systems).**

Regarding claim 5:

Zimmerman discloses the method as claimed in claim 1, wherein the data of the emulated hard disk or disks are accessible to the client station via a data processing network and wherein the emulated hard disk comprises one or more emulated hard disks **([0020]: network; [0022] and [0028]: multiple clients each with an emulated local disk).**

Regarding claim 6:

Zimmerman discloses the method as claimed in claim 1, wherein when an emulated hard disk is started up, the method further comprises a low level micro-software module to access data contained in the emulated hard disk wherein the operating system is started up at the client station **([0020]; [0058] microinstruction code).**

Regarding claim 7:

Zimmerman discloses the method as claimed in claim 6, wherein the data processing network comprises a plurality of client stations, **([0022]: multiple client devices),** the client stations using a bootup

PROM **([0024]: option PROM)**, the method further comprising using the bootup PROM to control communications via the data processing network **([0045]: communicate using PXE service).**

Regarding claim 8:

Zimmerman teaches the method as claimed in claim 7 wherein the low level micro-software module is loaded in the memory of the client station and executed using PROM **([0020]: microinstruction code; [0026] downloading of code).**

Regarding claim 9:

Zimmerman teaches the method as claimed in claim 6, wherein the low level micro software module is loaded in memory of the client station and executed as a component of BIOS of the client station, said micro software module providing the same functions as the access serviced on real hard disks provided by the BIOS **([0020]: microinstruction code; [0026] downloading of code; [0047]: BIOS).**

Regarding claim 10:

Zimmerman discloses a method as claimed in claim 6, wherein the low-level micro-software is loaded in memory of the client station from a third party data support supported as a startup peripheral by the client station **([0020]: microinstruction code; [0026] downloading of code).**

Regarding claim 11:

Zimmerman discloses the method as claimed in claim 5, wherein at least one peripheral driver loaded and executed by the operating system of the client station provided the functions of access, via the data processing network, to the data contained in the emulated hard disks **([0044]: storage driver and network filter driver used to download files from the virtual drive).**

Regarding claim 12:

Zimmerman discloses the method as claimed in claim 1 wherein if the data support does not provide for writing in real time, or does not to accept the writing of data directly in the data of the hard disk, the data writing requests issued by the operating system to the emulated hard disks are processed in a way such that the written data are stored in a storage space different from the data support containing the data of the emulated hard disks **([0069]: writes cached locally to prevent corruption of data)** and wherein the emulated hard disk comprises one or more emulated hard disks **([0022] and [0028]: multiple clients each with an emulated local disk))**

Regarding claim 13:

Zimmerman discloses the method as claimed in claim 12 wherein the data writing requests issued by the client station operating system to the emulated hard disks are processed in such a way that the written data are stored in the RAM of the client station **([0069]: writes cached locally to prevent corruption of data).**

Regarding claim 14:

Zimmerman discloses the method as claimed in claim 12, wherein the data writing requests issued by the client station operating system to the emulated hard disks are processed in such a way that the written data are stored in the virtual memory of the client station **([0069]: writes cached locally to prevent corruption of data).**

Regarding claim 15:

Zimmerman discloses the method as claimed in claim 12 wherein the data writing requests issued by the client station operating system to the emulated hard disks are processed in such a way that the written data are stored in a data file accessible to the operating system of the client station **([0069]: writes cached locally to prevent corruption of data).**

Regarding claim 16:

Zimmerman discloses the method as disclosed in claim 1 wherein the data writing requests issued by the operating system are redirected to a single storage space, and wherein the storage space in which the written data are redirected may be changed during an operating session of the operating system of a client station **([0021]: client module).**

Regarding claim 17:

Zimmerman discloses the method claimed in claim 12, wherein the storage space used for the storage may be volatile, or nonvolatile so as to permit the written data of an operating session of the operating system to persist from one client station to another **([0022]: client includes local memory, ROM, RAM, local storage).**

Regarding claim 18:

Zimmerman discloses the method as claimed in claim 1 wherein the volatile character of the redirections of the written data is determined upon initialization of the operating session of the operating system of a client station **([0021]: client module).**

Regarding claim 19:

Zimmerman discloses the method as claimed in claim 1, wherein the data reading requests issued by the operating system are performed in different storage spaces during an operating session of the OS of a client system **([0021]: multi-server environment).**

Regarding claim 20:

Zimmerman discloses the method as claimed in claim 19, wherein the data reading requests issued by the OS to an emulated hard disk carried out in different storage spaces follow an order of priority **([0062]).**

Regarding claim 21:

Zimmerman discloses the method as claimed in claim 5, wherein a server program is in charge at one of the client stations of the data processing network, on the one hand, of the communications via the data processing network with the client station accessing the emulated hard disks, and on the other, of accessing the data support containing the data of the emulated hard disks **([0026]: transparent emulation of operations of local disk at each of the clients; "virtual" storage device")**

Regarding claim 22:

Zimmerman discloses the method as claimed in claim 21, wherein if the hard disk emulation system is parameterized so that the data write requests received by the server program are intended for a specific emulated hard disk they are not redirected but stored directly in a support containing the data of the emulated hard disk itself and only client station can access said emulated hard disk at a given time **([0069]: storage driver caches locally all writes so that the writes are never committed to the virtual drive, in order that different clients to no simultaneously write to the same virtual image and corrupt it).**

Regarding claim 23:

Zimmerman discloses the method of claim 21, wherein in order to permit several client stations to access an emulated hard disk simultaneously, the server program is capable of redirecting specifically the data write requests issued by a client station to a given storage space and of redirecting the data write requests issued by another client station to another given storage space **([0027]: simultaneous access by different clients of either same or different data).**

Regarding claim 24:

Zimmerman discloses the method as in claim 21 wherein in order to permit the startup form and/or simultaneous access to the same emulated hard disk of 100% identical copies of the same emulated hard disk, components of the operating system loaded and executed by the client stations or server programs are capable of modifying during or before their use by the operating system data contained in the emulated hard disk **[0012]: software upgrades only performed on a single disk image**.

Regarding claim 25:

Zimmerman discloses performing emulation for the OS of the client stations at the level of the class of virtual peripherals of the file system type **([0026]: emulation).**

Regarding claim 26:

Zimmerman discloses performing the emulation for the OS at the level of the class of disk peripherals and not at the file system level **([0026]: emulation).**

Regarding claim 27:

Zimmerman discloses the method as claimed in claim 1, wherein the data contained in the emulated hard disk are copied by a software tool executed at a client station from the real hard disk **([0010]: download).**

Regarding claims 28 and 29:

Zimmerman discloses the method as claimed in claim 27 wherein the software tool creates an image file and directory that contains the data of the emulated hard disk **([0010]: downloading of image files).**

Regarding claim 30:

Rissmeyer teaches modifying the loading sequence so that all components of the OS are loaded and usable at the moment when the OS needs to access the emulated hard disk by the peripheral drivers **(column 1 lines 46-59: loading a network driver before loading a disk driver in an OS booting over a network).**

Regarding claim 31:

Zimmerman discloses the method as claimed in claim 21 wherein in order to accelerate the simultaneous access by several client stations to the same emulated hard disk whose data are contained in a data support accessible to a server station, the data are sent by the server station to the client stations using the "broadcast" or "multicast" mechanisms **([0010]: broadcast, multicast)**.

Regarding claim 32:

Zimmerman teaches storing the streamed data by the client station in a local cache situated in the memory of said client stations **([0032]: data stored in memory).**

Regarding claim 33:

Zimmerman teaches the method as claimed in claim 31 wherein a read request for data the in emulated hard disk issued by the operating system of a client station generates an explicit data reading request sent to the server station only if said data are not already present in the local cache **([0069]: reads read from virtual drive unless a copy of the data has already been cached on the client).**

Regarding claim 34:

Zimmerman teaches the method as claimed in claim 33 wherein the data in the local cache are removed after being read by the client station so as to free up space in said local cache **([0039]: cache emptied after data is no longer needed).**

Regarding claim 35:

Zimmerman teaches the method as claimed in claim 31 wherein the decision to send data by multicast/broadcast is made at the server module level which provides the functionalities necessary for the hard disk emulation at the client stations **([0010]: server performs broadcast, multicast).**

Regarding claim 36:

Zimmerman discloses the method as claimed in claim 31, wherein the client stations may modify their subscription to receiving the data sent via broadcast/multicast by the server station **([0038]: can choose to send data to only one client).**

Regarding claim 37:

Zimmerman discloses the method as claimed in claim 32, wherein the client stations may erase the data form the local cache after a certain parameterizable time **([0039]: cache emptied after data no longer needed).**

Regarding claim 38:

Zimmerman discloses the method as claimed in claim 5, wherein data contained in a server module making the hard disks available to client stations may use any suitable network protocol **([0020]: network). Rissmeyer teaches** using the iSCSI protocol **(abstract).**

Regarding claim 39:

Zimmerman discloses the method as claimed in claim 5, wherein the low level software program executed by the client stations and permitting access to the data contained in the emulated hard disks may use any suitable network protocol **([0020]: network). Rissmeyer teaches** using the iSCSI protocol **(abstract).**

Regarding claim 40:

Zimmerman discloses the method as claimed in claim 11, wherein at least one peripheral driver may use any suitable network protocol **([0020]: network). Rissmeyer teaches** using the iSCSI protocol **(abstract).**

Regarding claim 41:

Zimmerman discloses the method as claimed in claim 21 wherein if the data storage support does not provide writing in real time, or does not accept the writing of data directly in the data of the hard disk, the server program providing the emulation of the hard disk at the client stations processes the data write requests issued by the operating system in such a way that the written data are stored in a storage space different from the data support containing the data of the emulated hard disks **([0069]: writes cached locally to prevent corruption of data).**

Regarding claim 42:

Zimmerman discloses the method of claim 21 wherein the data write requests issued by the client station operating system to the emulated hard disks are processed in such a way that the written data are stored in the random access memory of the server station **([0069]: writes cached locally to prevent corruption of data).**

Regarding claim 43:

Zimmerman discloses the method as claimed in claim 21, wherein the data writing requests issued by the client station operating system to the emulated hard disks are processed in such a way that the written data are stored in the virtual memory of the client station **([0069]: writes cached locally to prevent corruption of data).**

Regarding claim 44:

Zimmerman discloses the method as claimed in claim 21 wherein the data writing requests issued by the client station operating system to the emulated hard disks are processed in such a way that the written data are stored in a data file accessible to the operating system of the client station **([0069]: writes cached locally to prevent corruption of data).**

Regarding claim 45:

Zimmerman discloses the method claimed in claim 21, wherein the storage space used for the storage may be volatile, or maybe be nonvolatile so as to permit the written data of an operating session of the operating system to persist from one client station to another **([0022]: client includes local memory, ROM, RAM, local storage).**

Regarding claim 46:

Zimmerman discloses the method as claimed in claim 16 wherein the volatile character of the redirections of the written data is determined upon initialization of the operating session of the operating system of a client station **([0021]: client module).**

(10) Response to Argument

(10.1) Regarding claim 1, Appellant submits on pages 6-8 of the appeal brief that Rissmeyer does not disclose "creating a representation of a real hard disk, wherein the *sequence*...for loading and execution of components of the operating system of the data processing platform may be modified" because Rissmeyer discloses a fixed order of initially loading a network driver and then loading a disk driver, and does not disclose that this predetermined loading of components may be modified.

Examiner's Answer:

Examiner notes that Appellants appear to be arguing limitations that are not apparent from the claims. Specifically, Appellant argues that Rissmeyer does not disclose the claimed modification of the sequence because the reference discloses a fixed order of loading the components. However, the reference discloses that the operating system boot sequence of the invention is different from the normal boot sequence of most other operating systems - because the operating system disk driver relies upon a network

driver, "…the traditional boot order of operating systems such as Microsoft Windows must be changed" **(see column 3 lines 3-14).** The reference then states that the order is modified such that a network driver is loaded before the system disk driver **(see column 3 lines 9-14).** This is sufficient to read on the claims, because the claim only requires a teaching of a one-time modification of the sequence **(see 1st limitation: "wherein the sequence…for loading and execution of components…may be modified)**, not a teaching that loading of the components be dynamically modifiable.

In other words, the Rissmeyer reference takes a loading sequence A (the normal Windows loading sequence) and modifies it to a fixed loading sequence of B **(see column 3 lines 3-14)**. Appellant argues that this does not disclose the claimed modified loading sequence because loading sequence B of the Rissmeyer reference is fixed (i.e. it cannot be further modified). Appellant appears to be arguing that the a predetermined order or sequence of loading and execution of components of an operating system changes or may be modified *after* the real hard disk representation is created. However, the claims only require that they be modified/changed *in order to create* the real hard disk representation.

(10.2) Regarding claim 1, Appellant submits on pages 8-9 of the appeal brief that Zimmerman does not disclose "creating a representation of a real hard disk, wherein the… *location* for loading and execution of components of the operating system…may be modified" because Zimmerman "…merely discloses that the 'address of the server to boot

from' is provided to a client 2 PC upon power-up…", but does not disclose that this location may be modified.

Examiner's Answer:

Paragraph [0058] of the Zimmerman reference states that the address of the server to boot from may or may not be the network server. This indicates that the address from which the components are to be loaded may be the address of the network server in one instance, and may be the address of a different server is a second instance. This interpretation is supported by **paragraph [0021] of the reference**, which states that in a multi-server network, the client may disclose to the client devices which additional server contains downloading information. Therefore, the reference teaches that the server from which the components are to be downloaded (i.e. loaded) is not static or fixed. Instead, when the client device requests information, the server responds with the address from which the information is to be downloaded **([0058]).** Therefore, the address is modified from one instance to the next based on which server has the requested information.

(10.3) Regarding claim 1, Appellant submits on page 10 of the appeal brief that the preamble discloses that the software emulation has parameterizable management of requests for reading and writing data, while Zimmerman has no mention of parameterization.

Examiner's Answer:

Examiner notes that by definition, read and write requests are parameterizable, wherein for a read request, the relevant parameter would be what data is requested to be

read (or the address from which the data is to be read), and wherein for a write request, the relevant parameter would be what data is requested to be written (or the address to which the data is to be written). While the Zimmerman reference does not specifically use the term "parameterizable", the reference nonetheless discloses parameterized read and write requests. Specifically, in **paragraph [0028]**, the reference states that each client desiring to download particular data issues an initial request (i.e. a read request), wherein the desired data may comprise any application files, O/S files, boot programs, etc. The specification of which particular data is requested to be read is a parameter, thus making the request parameterized. **Paragraphs [0035]-[0038]** further detail parameterized read requests: If a client has missed a previously streamed packet, the client will send a message indicating a need for retransmission of missing packets, and the needed data packets may be transmitted to the clients. Here, the specification of which packets are missing and need to be retransmitted comprises a parameterized read request. **Paragraph [0069]** further details parameterized read requests, wherein the read requests includes a "needed sector" (i.e. the data to be read), which is a parameter. Similarly, **paragraph [0069]** discloses parameterized write requests, wherein both the data to be written, and the location to which the data to be written are analogous to parameters (the writes are written to the local cache so that different clients do not simultaneously write to the same image).

(10.4) Regarding claims 2 and 41, Appellant submits on pages 11-12 of the appeal brief that the claims require that if the data support does not provide for writing in real

time or does not accept writing of data directly, then written data are stored in a storage space different from the data support, but in Zimmerman, the client may write to the server or to a local cache to avoid possible corruption of data, but not in response to the serve being unable to accept writing in real time or directly.

Examiner's Answer:

Examiner notes that as per **paragraph [0069],** all write requests are queued (i.e. no real-time writing requests are accepted). Specifically, when a write request is made to the OS disk, the request is not immediately sent to the disk -- instead, the write request is diverted to a cache in the client system's memory **(paragraph [0067]).** These requests are then, t a later time, sent to the O/S disk, and the data is then returned to the O/S **(paragraph [0067]).** Therefore, the server does not accept write requests as they are made(i.e. real-time/directly) in order to prevent different clients from simultaneously writing to the same image and corrupting it **(paragraph [0069]).** Appellant argues that the client may write to the server or to a local cache to avoid possible corruption of data, but not in response to the serve being unable to accept writing in real time or directly. However, the server does not accept direct write requests in order to avoid possible corruption of data – this is sufficient to read on the claimed limitations.

(10.5) Regarding claims 16 and 19, Appellant submits on pages 12-13 of the appeal brief that Zimmerman does not disclose changing to a different storage space for read or write requests *during an operating session*, as required by the claims.

Examiner's Answer:

The claim does not define or limit the term "operating session", and the term is interpreted to be analogous to the entire streaming process in figure 4, as well as subsequent retransmissions and booting processes. As disclosed in **paragraph [0019]**, the network server, which is where the disk is stored, may be implemented in a multi-server system. When a read request to an I/O disk is made, the location of the requested data may be on any of the servers, thereby disclosing that the location of the read request is performed in different storage spaces. Therefore, different read requests from different clients would result in different locations as the source of the data; accordingly, each client would access a different location for the requested data, which is analogous to a different storage space for read requests during an operation session. In **paragraph [0069],** the reference states that read requests are performed from the virtual drive, unless a copy of the needed sector has already been cached on the client. This is also analogous to a different storage space for read requests during an operation session, because when a read request is made, it is performed locally if the information is cached, and from the virtual drive if the information is not cached, thus resulting in a different storage space for the same request (i.e. operating session). Similarly, the reference teaches different storage spaces for write requests for an operation session in **paragraph [0032]**, wherein network and storage divers will load (i.e. write) the received data into a data cache that has been pre-allocated on <u>each</u> client (wherein said clients are described in **[0021]**) - therefore, each client has its own (i.e. different) cache to which the data is written during the streaming process (i.e. operating session).

(10.6) Regarding claim 20, Appellant submits on pages 13-15 of the appeal brief that the claim requires "wherein the data reading requests issued by the operating system to an emulated hard disk carried out in *different storage spaces* follow an *order of priority*", but Zimmerman discloses that a *single* network server streams the data packets based on an invitation period wherein the server designates a first client that will be allowed to make read requests, which cannot be characterized as the recited *order of priority*.

Examiner's Answer:

Paragraph [0062] details a read request process that follows an order of priority. The streaming module registers all clients which make an read request during an invitation period, and then designates a first client as a Read Requestor. Only the client designated as Read Requester will be allowed to make a read request (i.e. is given priority) of the streaming module to transmit to all the registered clients the contents of the next data sector. A different client will then be designated as Read Requestor, and will be allowed to make a read request (i.e. will be given priority) for a different data sector. Because different data sectors are necessarily stored at different addresses (even if they are all within the same drive), these constitute different storage spaces. Therefore, the reference discloses reading requests carried out in different storage spaces follow an order of priority.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

/Shambhavi Patel/

Examiner, Art Unit 2128

Conferees:

/Eddie C. Lee/

Quality Assurance Specialist, TC 2100

/KAMINI S SHAH/

Supervisory Patent Examiner, Art Unit 2128